High Fidelity Neural Audio Compression

Anonymous authors
Paper under double-blind review

Abstract

We introduce a state-of-the-art real-time, high-fidelity, audio codec leveraging neural networks. It consists in a streaming encoder-decoder architecture with quantized latent space trained in an end-to-end fashion. We simplify and speed-up the training by using a single multiscale spectrogram adversary that efficiently reduces artifacts and produce high-quality samples. We introduce a novel loss balancer mechanism to stabilize training: the *weight* of a loss now defines the fraction of the overall gradient it should represent, thus decoupling the choice of this hyper-parameter from the typical scale of the loss. Finally, we study how lightweight Transformer models can be used to further compress the obtained representation by up to 40%, while staying faster than real time. We provide a detailed description of the key design choices of the proposed model including: training objective, architectural changes and a study of various perceptual loss functions. We present an extensive subjective evaluation (MUSHRA tests) together with an ablation study for a range of bandwidths and audio domains, including speech, noisy-reverberant speech, and music. Our approach is superior to the baselines methods across all evaluated settings, considering both 24 kHz monophonic and 48 kHz stereophonic audio. Code and samples are available in the supplementary material.

1 Introduction

Recent studies suggest that streaming audio and video have accounted for the majority of the internet traffic in 2021 (82% according to (Cisco, 2021)). With the internet traffic expected to grow, audio compression is an increasingly important problem. In lossy signal compression we aim at minimizing the bitrate of a sample while also minimizing the amount of distortion according to a given metric, ideally correlated with human perception. Audio codecs typically employ a carefully engineered pipeline combining an encoder and a decoder to remove redundancies in the audio content and yield a compact bitstream. Traditionally, this is achieved by decomposing the input with a signal processing transform and trading off the quality of the components that are less likely to influence perception. Leveraging neural networks as trained transforms via an encoder-decoder mechanism has been explored by Morishima et al. (1990); Rippel et al. (2019); Zeghidour et al. (2021). Our research work is in the continuity of this line of work, with a focus on audio signals.

The problems arising in lossy neural compression models are twofold: first, the model has to represent a wide range of signals, such as not to overfit the training set or produce artifact laden audio outside its comfort zone. We solve this by having a large and diverse training set (described in Section 4.1), as well as discriminator networks (see Section 3.4) that serve as perceptual losses, which we study extensively in Section 4.5.1, Table 2. The other problem is that of compressing efficiently, both in compute time and in size. For the former, we limit ourselves to models that run in real-time on a single CPU core. For the latter, we use residual vector quantization of the neural encoder floating-point output, for which various approaches have been proposed (Van Den Oord et al., 2017; Zeghidour et al., 2021).

Accompanying those technical contributions, we posit that designing end-to-end neural compression models is a set of intertwined choices, among which at least the encoder-decoder architecture, the quantization method, and the perceptual loss play key parts. Objective evaluations exist and we report scores on them in our ablations (Section 4.5.1). But the evaluation of lossy audio codecs necessarily relies on human perception, so we ran extensive human evaluation for multiple points in this design space, both for speech and music. Those evaluations (MUSHRA) consist in having humans listen to, compare, and rate excerpts of speech or music

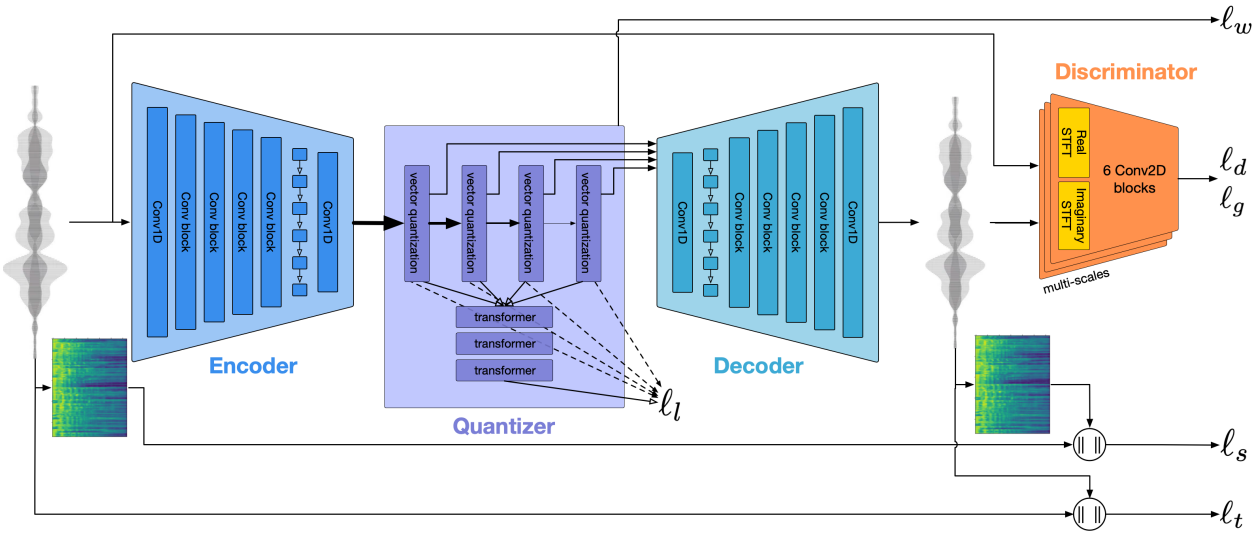

Figure 1: EnCodec : an encoder decoder codec architecture which is trained with reconstruction (ℓ_f and ℓ_t) as well as adversarial losses (ℓ_g for the generator and ℓ_d for the discriminator). The residual vector quantization commitment loss (ℓ_w) applies only to the encoder. Optionally, we train a small Transformer language model for entropy coding over the quantized units with ℓ_l, which reduces bandwidth even further.

compressed with competitive codecs and variants of our method, and the uncompressed ground truth. This allows to compare variants of the whole pipeline in isolation, as well as their combined effect, in Section 4.5.1 (Figure 3 and Table 1). Finally, our best model, EnCodec , reaches state-of-the-art scores for speech and for music at 1.5, 3, 6, 12 kbps at 24 kHz, and at 6, 12, and 24 kbps for 48 kHz with stereo channels.

2 Related Work

Speech and Audio Synthesis. Recent advancements in neural audio generation enabled computers to efficiently generate natural sounding audio. The first convincing results were achieved by autoregressive models such as WaveNet (Oord et al., 2016), at the cost of slow inference. While many other approaches were explored (Yamamoto et al., 2020a; Kalchbrenner et al., 2018; Goel et al., 2022), the most relevant ones here are those based on Generative Adversarial Networks (GAN) (Kumar et al., 2019; Yamamoto et al., 2020a; Kong et al., 2020; Andreev et al., 2022) were able to match the quality of autoregressive by combining various adversarial networks operate at different multi-scale and multi-period resolutions. Our work uses and extends similar adversarial losses to limit artifacts during audio generation.

Audio Codec. Low bitrate parametric speech and audio codecs have long been studied (Atal & Hanauer, 1971; Juang & Gray, 1982), but their quality has been severely limited. Despite some advances (Griffin & Lim, 1985; McCree et al., 1996), modeling the excitation signal has remained a challenging task. The current state-of-the-art traditional audio codecs are Opus (Valin et al., 2012) and Enhanced Voice Service (EVS) (Dietz et al., 2015). These methods produce high coding efficiency for general audio while supporting various bitrates, sampling rates, and real-time compression.

Neural based audio codecs have been recently proposed and demonstrated promising results (Kleijn et al., 2018; Valin & Skoglund, 2019b; Lim et al., 2020; Kleijn et al., 2021; Zeghidour et al., 2021; Omran et al., 2022; Lin et al., 2022; Jayashankar et al., 2022; Li et al.; Jiang et al., 2022), where most methods are based on quantizing the latent space before feeding it to the decoder. In Valin & Skoglund (2019b), an LPCNet (Valin & Skoglund, 2019a) vocoder was conditioned on hand-crafted features and a uniform quantizer. Gârbacea et al. (2019) conditioned a WaveNet based model on discrete units obtained from a VQ-VAE (Van Den Oord et al., 2017; Razavi et al., 2019) model, while Skoglund & Valin (2019) tried feeding the Opus codec (Valin et al., 2012) to a WaveNet to further improve its perceptual quality. Jayashankar et al. (2022); Jiang et al. (2022) propose an auto-encoder with a vector quantization layer applied over the latent representation and minimizing the

reconstruction loss, while Li et al. suggested using Gumbel-Softmax (GS) (Jang et al., 2017) for representation quantization. The most relevant related work to ours is the SoundStream model (Zeghidour et al., 2021), in which the authors propose a fully convolutional encoder decoder architecture with a Residual Vector Quantization (RVQ) (Gray, 1984; Vasuki & Vanathi, 2006) layers. The model was optimized using both reconstruction loss and adversarial perceptual losses. Caillon & Esling (2021) studied compression as part of VAE-based audio modeling , but they did not report any objective or subjective evaluations for this particular application.

Audio Discretization. Representing audio and speech using discrete values was proposed to various tasks recently. Dieleman et al. (2018); Dhariwal et al. (2020) proposed a hierarchical VQ-VAE based model for learning discrete representation of raw audio, next combined with an auto-regressive model, demonstrating the ability to generate high quality music. Similarly, Lakhotia et al. (2021); Kharitonov et al. (2021) demonstrated that self-supervised learning methods for speech (e.g., HuBERT (Hsu et al., 2021)), can be quantized and used for conditional and unconditional speech generation. Similar methods were applied to speech resynthesis (Polyak et al., 2021), speech emotion conversion (Kreuk et al., 2021), spoken dialog system (Nguyen et al., 2022), and speech-to-speech translation (Lee et al., 2021a;b; Popuri et al., 2022).

3 Model

An audio signal of duration d can be represented by a sequence $\boldsymbol{x} \in [-1, 1]^{C_\mathrm{a} \times T}$ with C_a the number of audio channels, $T = d \cdot f_\mathrm{sr}$ the number of audio samples at a given sample rate f_sr. The EnCodec model is composed of three main components: (i) First, an encoder network E is input an audio extract and outputs a latent representation $\boldsymbol{z}$; (ii) Next, a quantization layer Q produces a compressed representation $\boldsymbol{z}_q$, using vector quantization; (iii) Lastly, a decoder network G reconstructs the time-domain signal, $\hat{\boldsymbol{x}}$, from the compressed latent representation $\boldsymbol{z}_q$. The whole system is trained end-to-end to minimize a reconstruction loss applied over both time and frequency domain, together with a perceptual loss in the form of discriminators operating at different resolutions. A visual description of the proposed method can be seen in Figure 1.

3.1 Encoder & Decoder Architecture

The EnCodec model is a simple streaming, convolutional-based encoder-decoder architecture with sequential modeling component applied over the latent representation, both on the encoder and on the decoder side. Such modeling framework was shown to provide great results in various audio-related tasks, e.g., source separation and enhancement (Défossez et al., 2019; Defossez et al., 2020), neural vocoders (Kumar et al., 2019; Kong et al., 2020), audio codec (Zeghidour et al., 2021), and artificial bandwidth extension (Tagliasacchi et al., 2020; Li et al., 2021). We use the same architecture for 24 kHz and 48 kHz audio.

Encoder-Decoder. The encoder model E consists in a 1D convolution with C channels and a kernel size of 7 followed by B convolution blocks. Each convolution block is composed of a single residual unit followed by a down-sampling layer consisting in a strided convolution, with a kernel size K of twice the stride S. The residual unit contains two convolutions with kernel size 3 and a skip-connection. The number of channels is doubled whenever down-sampling occurred. The convolution blocks are followed by a two-layer LSTM for sequence modeling and a final 1D convolution layer with a kernel size of 7 and $D = 128$ output channels. Following Zeghidour et al. (2021); Li et al. (2021), we use $C = 32$, $B = 4$ and (2, 4, 5, 8) as strides. We use ELU as a non-linear activation function (Clevert et al., 2015) either layer normalization (Ba et al., 2016) or weight normalization (Salimans & Kingma, 2016). We use two variants of the model, depending on whether we target the low-latency streamable setup, or a high fidelity non-streamable usage. With this setup, the encoder outputs 75 latent steps per second of audio at 24 kHz, and 150 at 48 kHz. The decoder mirrors the encoder, using transposed convolutions instead of strided convolutions, and with the strides in reverse order as in the encoder, outputting the final mono or stereo audio.

Non-streamable. In the non-streamable setup, we use for each convolution a total padding of $K - S$, split equally before the first time step and after the last one (with one more before if $K - S$ is odd). We further split the input into chunks of 1 seconds, with an overlap of 10 ms to avoid clicks, and normalize each chunk before feeding it to the model, applying the inverse operation on the output of the decoder, adding a

negligible bandwidth overhead to transmit the scale. We use layer normalization (Ba et al., 2016), computing the statistics including also the time dimension in order to keep the relative scale information.

Streamable. For the streamable setup, all padding is put before the first time step. For a transposed convolution with stride s, we output the s first time steps, and keep the remaining s steps in memory for completion when the next frame is available, or discarding it at the end of a stream. Thanks to this padding scheme, the model can output 320 samples (13 ms) as soon as the first 320 samples (13 ms) are received. We replace the layer normalization with statistics computed over the time dimension with weight normalization (Salimans & Kingma, 2016), as the former is ill-suited for a streaming setup. We notice a small gain over the objective metrics by keeping a form of normalization, as demonstrated in Table A.3.

3.2 Residual Vector Quantization

We use Residual Vector Quantization (RVQ) to quantize the output of the encoder as introduced by Zeghidour et al. (2021). Vector quantization consists in projecting an input vector onto the closest entry in a codebook of a given size. RVQ refines this process by computing the residual after quantization, and further quantizing it using a second codebook, and so forth.

We follow the same training procedure as described by Dhariwal et al. (2020) and Zeghidour et al. (2021). The codebook entry selected for each input is updated using an exponential moving average with a decay of 0.99, and entries that are not used are replaced with a candidate sampled from the current batch. We use a straight-through-estimator (Bengio et al., 2013) to compute the gradient of the encoder, e.g. as if the quantization step was the identity function during the backward phase. Finally, a commitment loss, consisting of the MSE between the input of the quantizer and its output, with gradient only computed with respect to its input, is added to the overall training loss.

By selecting a variable number of residual steps at train time, a single model can be used to support multiple bandwidth target (Zeghidour et al., 2021). For all of our models, we use at most 32 codebooks (16 for the 48 kHz models) with 1024 entries each, e.g. 10 bits per codebook. When doing variable bandwidth training, we select randomly a number of codebooks as a multiple of 4, i.e. corresponding to a bandwidth 1.5, 3, 6, 12 or 24 kbps at 24 kHz. Given a continuous latent represention with shape $[B, D, T]$ that comes out of the encoder, this procedure turns it into a discrete set of indexes $[B, N_q, T]$ with N_q the number of codebooks selected. This discrete representation can changed again to a vector by summing the corresponding codebook entries, which is done just before going into the decoder.

3.3 Language Modeling and Entropy Coding

We additionally train a small Transformer based language model (Vaswani et al., 2017) with the objective of keeping faster than real time end-to-end compression/decompression on a single CPU core. The model consists of 5 layers, 8 heads, 200 channels, a dimension of 800 for the feed-forward blocks, and no dropout. At train time, we select a bandwidth and the corresponding number of codebooks N_q. For a time step t, the discrete representation obtained at time $t-1$ is transformed into a continuous representation using learnt embedding tables, one for each codebook, and which are summed. For $t = 0$, a special token is used instead. The output of the Transformer is fed into N_q linear layers with as many output channels as the cardinality of each codebook (e.g. 1024), giving us the logits of the estimated distribution over each codebook for time t. We thus neglect potential mutual information between the codebooks at a single time step. This allows to speedup inference (as opposed to having one time step per codebook, or a multi-stage prediction) with a limited impact over the final cross entropy. Each attention layer has a causal receptive field of 3.5 seconds, and we offset by a random amount the initial position of the sinusoidal position embedding to emulate being in a longer sequence. We train the model on sequences of 5 seconds.

Entropy Encoding. We use a range based arithmetic coder (Pasco, 1976; Rissanen & Langdon, 1981) in order to leverage the estimated probabilities given by the language model. As noted by Ballé et al. (2018), evaluation of the same model might lead to different results on different architectures, or with different evaluation procedures due to floating point approximations. This can lead to decoding errors as the encoder and decoder will not use the exact same code. We observe in particular that the difference between batch

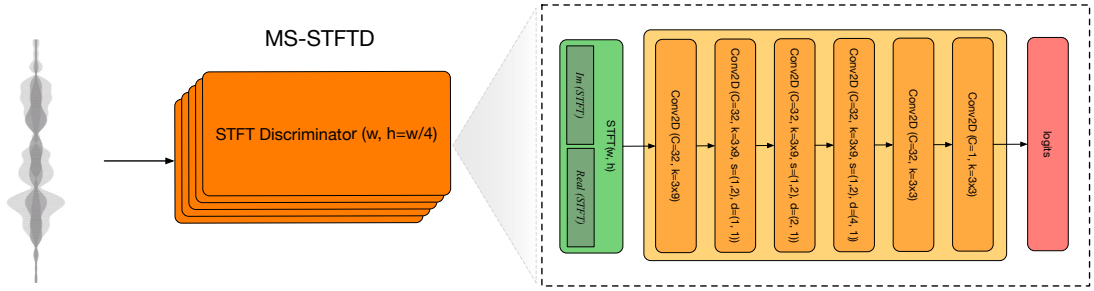

Figure 2: MS-STFT Discriminator architecture. The input to the network is a complex-valued STFT with the real and imaginary parts concatenated. Each discriminator is composed of a 2D convolutional layer, followed by 2D convolutions with increasing dilation rates. Then a final 2D convolution is applied.

evaluation (e.g. all time steps at once), and the real-life streaming evaluation that occurs in the decoder can lead to difference larger than 10^{-8}. We first round the estimated probabilities with a precision of 10^{-6}, although evaluations in more contexts would be needed for practical deployment. We use a total range width of 2^{24}, and assign a minimum range width of 2. We discuss the impact on the processing time in Section 4.6.

3.4 Training objective

We detail the training objective that combines a reconstruction loss term, a perceptual loss term (via discriminators), and the RVQ commitment loss.

Reconstruction Loss. The reconstruction loss term is comprised of a time and a frequency domain loss term. We minimize the L1 distance between the target and compressed audio over the time domain, i.e. $\ell_t(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_1$. For the frequency domain, we use a linear combination between the L1 and L2 losses over the mel-spectrogram using several time scales (Yamamoto et al., 2020b; Gritsenko et al., 2020). Formally,

$$\ell_f(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{|\alpha| \cdot |s|} \sum_{\alpha_i \in \alpha} \sum_{i \in e} \|\mathcal{S}_i(\boldsymbol{x}) - \mathcal{S}_i(\hat{\boldsymbol{x}})\|_1 + \alpha_i \|\mathcal{S}_i(\boldsymbol{x}) - \mathcal{S}_i(\hat{\boldsymbol{x}})\|_2, \tag{1}$$

where $\mathcal{S}_i$ is a 64-bins mel-spectrogram using a normalized STFT with window size of 2^i and hop length of $2^i/4$, $e = 5, \ldots, 11$ is the set of scales, and α represents the set of scalar coefficients balancing between the L1 and L2 terms. Unlike Gritsenko et al. (2020), we take $\alpha_i = 1$.

Discriminative Loss. To further improve the quality of the generated samples, we introduce a perceptual loss term based on a multi-scale STFT-based (MS-STFT) discriminator, illustrated in Figure 2. Multi scale discriminators are popular for capturing different structures in audio signals (Kumar et al., 2019; Kong et al., 2020; You et al., 2021). The MS-STFT discriminator consists in identically structured networks operating on multi-scaled complex-valued STFT with the real and imaginary parts concatenated. Each sub-network is composed of a 2D convolutional layer (using kernel size 3 x 8 with 32 channels), followed by 2D convolutions with increasing dilation rates in the time dimension of 1, 2 and 4, and a stride of 2 over the frequency axis. A final 2D convolution with kernel size 3 x 3 and stride (1, 1) provide the final prediction. We use 5 different scales with STFT window lengths of [2048, 1024, 512, 256, 128]. For 48 kHz audio, we double the size of each STFT window and train the discriminator every two batches, and for stereophonic audio, we process separately the left and right channels. We use LeakyReLU as a non-linear activation function and apply weight normalization (Salimans & Kingma, 2016) to our discriminator network. The MS-STFT discriminator model architecture is visually depicted in Figure 2.

The adversarial loss for the generator is constructed as follows, $\ell_g(\hat{\boldsymbol{x}}) = \frac{1}{K} \sum_k \max(0, 1 - D_k(\hat{\boldsymbol{x}})))$, where K is the number of discriminators. Similarly to previous work on neural vocoders (Kumar et al., 2019; Kong et al., 2020; You et al., 2021), we additionally include a relative feature matching loss for the generator. Formally,

$$\ell_{feat}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{KL} \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{\|D_k^l(\boldsymbol{x}) - D_k^l(\hat{\boldsymbol{x}})\|_1}{\text{mean}\left(\|D_k^l(\boldsymbol{x})\|_1\right)}, \tag{2}$$

where the mean is computed over all dimensions, (D_k) are the discriminators, and L is the number of layers in discriminators. The discriminators are trained to minimize the following hinge-loss adversarial loss function: $L_d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{K}\sum_{k=1}^{K} \max(0, 1 - D_k(\boldsymbol{x})) + \max(0, 1 + D_k(\hat{\boldsymbol{x}}))$, where K is the number of discriminators. Given that the discriminator tend to overpower easily the decoder, we update its weight with a probability of 2/3 at 24 kHz, and 0.5 at 48 kHz.

Multi-bandwidth training. At 24 kHz, we train the model to support the bandwidths 1.5, 3, 6, 12, and 24 kbps by selecting the appropriate number of codebooks to keep in the RVQ step, as explained in Section 3.2. At 48 kHz, we train to support 3, 6, 12 and 24 kbps. We also noticed that using a dedicated discriminator per-bandwidth is beneficial to the audio quality. Thus, we select a given bandwidth for the entire batch, and evaluate and update only the corresponding discriminator.

VQ commitment loss. As mentioned in Section 3.2, we add a commitment loss l_w between the output of the encoder, and its quantized value, with no gradient being computed for the quantized value. For each residual step $c \in \{1, \dots C\}$ (with C depeding on the bandwidth target for the current batch), noting $\boldsymbol{z}_c$ the current residual and $q_c(\boldsymbol{z}_c)$ the nearest entry in the corresponding codebook, we define l_w as

$$l_w = \sum_{c=1}^{C} \|\boldsymbol{z}_c - q_c(\boldsymbol{z}_c)\|_2^2. \tag{3}$$

Overall, the generator is trained to optimize the following loss, summed over the batch,

$$L_G = \lambda_t \cdot \ell_t(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_f \cdot \ell_f(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_g \cdot \ell_g(\hat{\boldsymbol{x}}) + \lambda_{feat} \cdot \ell_{feat}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda_w \cdot \ell_w(w), \tag{4}$$

where λ_t, λ_f, λ_g, λ_{feat}, and λ_w the scalar coefficients to balance between the terms.

Balancer. We introduce a loss balancer in order to stabilize training, in particular the varying scale of the gradients coming from the discriminators. We also find that the balancer makes it easier to reason about the different loss weights, independently of their scale. Let us take a number of losses $(\ell_i)_i$ that depends only on the output of the model $\hat{x}$. We define $g_i = \frac{\partial \ell_i}{\partial \hat{x}}$, and $\langle \|g_i\|_2 \rangle_\beta$ the exponential moving average of g_i over the last training batches. Given a set of weights (λ_i) and a reference norm R, we define

$$\tilde{g}_i = R \frac{\lambda_i}{\sum_j \lambda_j} \cdot \frac{g_i}{\langle \|g_i\|_2 \rangle_\beta}. \tag{5}$$

We then backpropagate into the network $\sum_i \tilde{g}_i$, instead of the original $\sum_i \lambda_i g_i$. This changes the optimization problem but allows to make the λ_i interpretable irrespectively of the natural scale of each loss. If $\sum_i \lambda_i = 1$, then each weight can be interpreted as the fraction of the model gradient that come from the corresponding loss. We take $R = 1$ and $\beta = 0.999$. All the generator losses from Eq. (4) fit into the balancer, except for the commitment loss, as it is not defined with respect to the output of the model.

4 Experiments and Results

4.1 Dataset

We train EnCodec on 24 kHz monophonic across diverse domains, namely: speech, noisy speech, music and general audio while we train the fullband stereo EnCodec on only 48 kHz music. For speech, we use the clean speech segments from DNS Challenge 4 (Dubey et al., 2022) and the Common Voice dataset (Ardila et al., 2019). For general audio, we use on AudioSet (Gemmeke et al., 2017) together with FSD50K (Fonseca et al., 2021). For music, we rely on the Jamendo dataset (Bogdanov et al., 2019) for training and evaluation and we further evaluate our models on music using a proprietary music dataset. Data splits are detailed in Appendix A.1.

For training and validation, we define a mixing strategy which consists in either sampling a single source from a dataset or performing on the fly mixing of two or three sources. Specifically, we have four strategies: (s1) we sample a single source from Jamendo with probability 0.32; (s2) we sample a single source from the other datasets with the same probability; (s3) we mix two sources from all datasets with a probability of 0.24; (s4) we mix three sources from all datasets except music with a probability of 0.12.

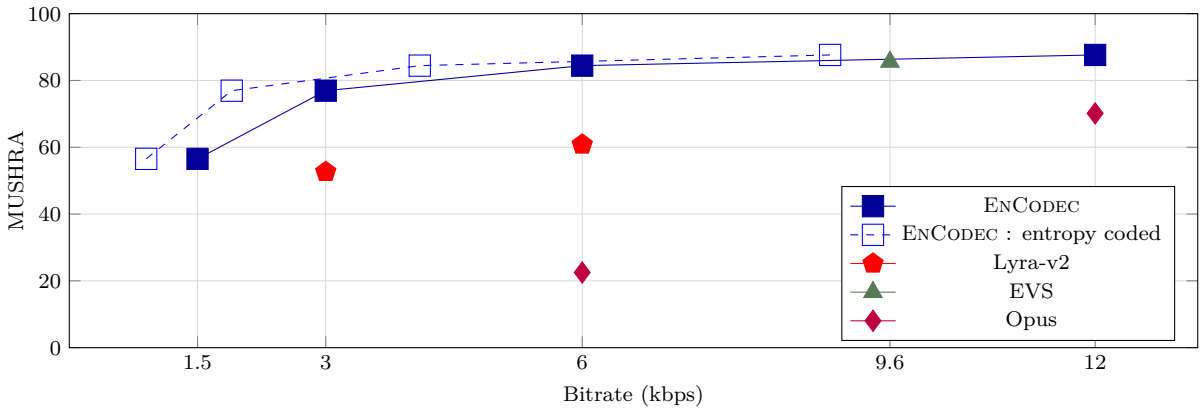

Figure 3: Human evaluations (MUSHRA: comparative scoring of samples) across bandwidths of standard codecs and neural codecs. For EnCodec we report the initial bandwidth without entropy coding (in plain) and with entropy coding (hollow). Lyra-v2 is a neural audio codec, while EVS and Opus are competitive standard codecs. The audio samples are from speech and music. The ground truth is 16bits 24kHz wave.

The audio is normalized by file and we apply a random gain between -10 and 6 dB. We reject any sample that has been clipped. Finally we add reverberation using room impulse responses provided by the DNS challenge with probability 0.2, and RT60 in the range [0.3, 1.3] except for the single-source music samples. For testing, we use four categories: clean speech from DNS alone, clean speech mixed with FSDK50K sample, Jamendo sample alone, proprietary music sample alone.

4.2 Baselines

Opus (Valin et al., 2012) is a versatile speech and audio codec standardized by the IETF in 2012. It scales from 6 kbps narrowband monophonic audio to 510 kbps fullband stereophonic audio. EVS (Dietz et al., 2015) is a codec standardized in 2014 by 3GPP and developed for Voice over LTE (VoLTE). It supports a range of bitrates from 5.9 kbps to 128 kbps, and audio bandwidths from 4 kHz to 20 kHz. It is the successor of AMR-WB (Bhagat et al., 2012).We use both codecs to serve as traditional digital signal processing baselines. We also utilize MP3 compression at 64 kbps as an additional baseline for the stereophonic signal compression case. MP3 uses lossy data compression by approximating the accuracy of certain components of sound that are considered to be beyond hearing capabilities of most humans. Finally, we compare EnCodec to the SoundStream model from the official implementation available in Lyra 2 [1] at 3.2 kbps and 6 kbps on audio upsampled to 32 kHz. We also reproduced a version of SoundStream (Zeghidour et al., 2021) with minor improvements. Namely, we use the relative feature loss introduce in Section 3.4, and layer normalization (applied separately for each time step) in the discriminators, except for the first and last layer, which improved the audio quality during our preliminary studies. Results a reported in Table A.2 in the Appendix A.3.

4.3 Evaluation Methods

We consider subjective and objective metrics. For the subjective tests we follow the MUSHRA protocol (Series, 2014), using a hidden reference and a low anchor. Annotators were recruited using a crowd-sourcing platform, in which they were asked to rate the perceptual quality of the provided samples in a range between 1 to 100. We randomly select 50 samples of 5 seconds from each category of the the test set and force at least 10 annotations per samples. To filter noisy annotations and outliers we remove annotators who rate the reference recordings less then 90 in at least 20% of the cases, or rate the low-anchor recording above 80 more than 50% of the time. For objective metrics, we use ViSQOL (Hines et al., 2012; Chinen et al., 2020) [2],

[1] https://github.com/google/lyra
[2] We compute visqol with: `https://github.com/google/visqol` using the recommended recipes.

Table 1: MUSHRA scores for Opus, EVS, Lyra-v2, and EnCodec for various bandwidths under the streamable setting. Results are reported across different audio categories (clean speech, noisy speech, and music), sampled at 24 kHz. We report mean scores and 95% confidence intervals. For EnCodec , we also report the average bandwidth after using the entropy coding described in Section 3.3.

Model	Bandwidth	Entropy Coded	Clean Speech	Noisy Speech	Music Set-1	Music Set-2
Reference	-	-	95.5$\pm$1.6	93.9$\pm$1.8	93.2$\pm$2.5	97.1$\pm$1.3
Opus	6.0 kbps	-	30.1$\pm$2.8	19.1$\pm$5.9	20.6$\pm$5.8	17.9$\pm$5.3
Opus	12.0 kbps	-	76.5$\pm$2.3	61.9$\pm$2.1	77.8$\pm$3.2	65.4$\pm$2.7
EVS	9.6 kbps	-	84.4$\pm$2.5	80.0$\pm$2.4	89.9$\pm$2.3	87.7$\pm$2.3
Lyra-v2	3.0 kbps	-	53.1$\pm$1.9	52.0$\pm$4.7	69.3$\pm$3.3	42.3$\pm$3.5
Lyra-v2	6.0 kbps	-	66.2$\pm$2.9	59.9$\pm$3.3	75.7$\pm$2.6	48.6$\pm$2.1
EnCodec	1.5 kbps	0.9 kbps	49.2$\pm$2.4	41.3$\pm$3.6	68.2$\pm$2.2	66.5$\pm$2.3
EnCodec	3.0 kbps	1.9 kbps	67.0$\pm$1.5	62.5$\pm$2.3	89.6$\pm$3.1	87.8$\pm$2.9
EnCodec	6.0 kbps	4.1 kbps	83.1$\pm$2.7	69.4$\pm$2.3	92.9$\pm$1.8	91.3$\pm$2.1
EnCodec	12.0 kbps	8.9 kbps	90.6$\pm$2.6	80.1$\pm$2.5	91.8$\pm$2.5	92.9$\pm$1.2

together with the Scale-Invariant Signal-to-Noise Ration (SI-SNR) (Luo & Mesgarani, 2019; Nachmani et al., 2020; Chazan et al., 2021).

4.4 Training

We train all models for 300 epochs, with one epoch being 2,000 updates with the Adam optimizer with a batch size of 64 examples of 1 second each, a learning rate of $3 \cdot 10^{-4}$, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. All the models are traind using 8 A100 GPUs. We use the balancer introduced in Section 3.4 with weights $\lambda_t = 0.1$, $\lambda_f = 1$, $\lambda_g = 3$, $\lambda_{\text{feat}} = 3$ for the 24 kHz models. For the 48 kHz model, we use instead $\lambda_g = 4$, $\lambda_{\text{feat}} = 4$.

4.5 Results

We start with the results for EnCodec with a bandwidth in $\{1.5, 3, 6, 12\}$ kbps and compare them to the baselines. Results for the streamable setup are reported in Figure 3 and a breakdown per category in Table 1. We additionally explored other quantizers such as Gumbel-Softmax and DiffQ (see details in Appendix A.2), however, we found in preliminary results that they provide similar or worse results, hence we do not report them.

When considering the same bandwidth, EnCodec is superior to all evaluated baselines considering the MUSHRA score. Notice, EnCodec at 3kbps reaches better performance on average than Lyra-v2 using 6kbps and Opus at 12kbps. When considering the additional language model over the codes, we can reduce the bandwidth by $\sim 25 - 40\%$. For instance, we can reduce the bandwidth of the 3 kpbs model to 1.9 kbps. We observe that for higher bandwidth, the compression ratio is lower, which could be explained by the small size of the Transformer model used, making hard to model all codebooks together.

4.5.1 Ablation study

Next, we perform an ablation study to better evaluate the effect of the discriminator setup, streaming, multi-target bandwidth, and balancer. We provide more detailed ablation studies in the Appendix, Section A.3.

The effect of discriminators setup. Various discriminators were proposed in prior work to improve the perceptual quality of the generated audio. The Multi-Scale Discriminator (MSD) model proposed by Kumar et al. (2019) and adopted in (Kong et al., 2020; Andreev et al., 2022; Zeghidour et al., 2021), operates on the raw waveform at different resolutions. We adopt the same MSD configuration as described in Zeghidour et al. (2021). Kong et al. (2020) additionally propose the Multi-Period Discriminator (MPD) model, which reshapes

Table 2: Comparing discriminators using objective (ViSQOL, SI-SNR) and subjective metrics (MUSHRA).

Discriminator setup	SI-SNR	ViSQOL	MUSHRA
MSD+Mono-STFT	5.99	4.22	62.91$\pm$2.62
MPD	7.35	4.24	60.7$\pm$2.8
MS-STFT+MPD	6.55	**4.34**	**79.0$\pm$1.9**
MS-STFT	**6.67**	**4.35**	**77.5$\pm$1.8**

Table 3: Streamable vs. Non-streamable evaluations at 6 kbps on an equal mix of speech and music.

Model	Streamable	SI-SNR	ViSQOL
Opus	✓	2.45	2.60
EVS	✓	1.89	2.74
EnCodec	✓	6.67	4.35
EnCodec	✗	7.46	4.39

the waveform to a 2D input with multiple periods. Next, the STFT Discriminator (Mono-STFTD) model was introduced in Zeghidour et al. (2021), where a single network operates over the complex-valued STFT.

We evaluate our MS-STFTD discriminator against three other discriminator configurations: (i) MSD+Mono-STFTD (as in Zeghidour et al. (2021)); (ii) MPD only; (iii) MS-STFTD only; (vi) MS-STFTD+MPD. Results are reported in Table 2. Results suggest that using only a multi-scale STFT-based discriminator such as MS-STFTD, is enough to generate high quality audio. Additionally, it simplifies the model training and reduces training time. Including the MPD discriminator, adds a small gain when considering the MUSHRA score.

The effect of the streamable modeling. We also investigate streamable vs. non-streamable setups and report results in Table 3. Unsurprisingly, we notice a small degradation switching from non-streamable to streamable but the performance remains strong while this setting enables streaming inference.

The effect of the balancer. Lastly, we present results evaluating the impact of the balancer. We train the EnCodec model considering various values λ_t, λ_f, λ_g, and λ_{feat} with and without the balancer. Results are reported in Table A.4 in the Appendix. As expected, results suggest the balancer significantly stabilizes the training process. See Appendix A.3 for more details.

4.5.2 Stereo Evaluation

All previously reported results considered only the monophonic setup. Although it makes sense when considering speech data, however for music data, stereo compression is highly important. We adjust our current setup to stereo by only modifying our discriminator setup as described in Section 3.4.

Table 4: **Stereophonic** extreme music compression versus MP3 and Opus for music sampled at 48 kHz.

Model	Bandwidth	Entropy Coded	Compression	MUSHRA
Reference	-	-	1×	**95.1$\pm$1.8**
MP3	64 kbps	-	24×	82.7$\pm$3.2
Opus	6 kbps	-	256×	17.7$\pm$5.9
Opus	24 kbps	-	64×	82.9$\pm$3.7
EnCodec	6 kbps	4.2 kbps	256×	82.9$\pm$2.4
EnCodec	12 kbps	8.9 kbps	128×	**88.0$\pm$2.7**
EnCodec	24 kbps	19.4 kbps	64×	**87.5$\pm$2.6**

Table 5: Initial latency and real time factor (RTF) for Lyra v2, EnCodec at 24 kHz and 48 kHz. A RTF greater than 1 indicates faster than real time processing. We report the RTF for both the encoding (Enc.) and decoding (Dec.), without and with entropy coding (EC). All models are evaluated at 6 kbps.

		Real Time Factor			
Model	Latency	*Enc.*	*Dec.*	*Enc. + EC*	*Dec. + EC*
Lyra v2 (32 kHz)	-	27.4	67.2	-	-
EnCodec 24 kHz	13 ms	9.8	10.4	1.6	1.6
EnCodec 48 kHz	1 s	6.8	5.1	0.68	0.66

Results for EnCodec working at 6 kbps, EnCodec with Residual Vector Quantization (RVQ) at 6 kbps, and Opus at 6 kbps, and MP3 at 64 kbps are reported in Table 4. EnCodec is significantly outperforms Opus at 6kbps and is comparable to MP3 at 64kbps, while EnCodec at 12kpbs achieve comparable performance to EnCodec at 24kbps. Using a language model and entropy coding gives a variable gain between 20% to 30%.

4.6 Latency and computation time

We report the initial latency and real time factor on Table 5. The real-time factor is here defined as the ratio between the duration of the audio and the processing time, so that it is greater than one when the method is faster than real time. We profiled all models on a single thread of a MacBook Pro 2019 CPU at 6 kbps.

Initial latency. The 24 kHz streaming EnCodec model has an initial latency (i.e., without the computation time) of 13.3 ms. The 48 kHz non-streaming version has an initial latency of 1 second, due to the normalizations used. Note that using entropy coding increases the initial latency, because the stream cannot be "flushed" with each frame, in order to keep the overhead small. Thus decoding the frame at time t, requires for the frame $t + 1$ to be partially received, increasing the latency by 13ms.

Real time factor. While our model is worse than Lyra v2 in term of processing speed, it processes the audio 10 times faster than real time, making it a good candidate for real life applications. The gain from the entropy coding comes at a cost, although the processing is still faster than real time and could be used for applications where latency is not essential (e.g. streaming). At 48 kHz, the increased number of step size lead to a slower than real time processing, although a more efficient implementation, or using accelerated hardware would improve the RTF. It could also be used for archiving where real time processing is not required.

5 Conclusion

We presented EnCodec : a state-of-the-art real-time neural audio compression model, producing high-fidelity audio samples across a range of sample rates and bandwidth. We showed subjective and objective results from 24kHz monophonic at 1.5 kbps (Figure 3) to 48kHz stereophonic (Table 4). We improved sample quality by developing a simple but potent spectrogram-only adversarial loss which efficiently reduces artifacts and produce high-quality samples. Besides, we stabilized training and improved the interpretability of the weights for losses through a novel gradient balancer. Finally, we also demonstrated that a small Transformer model can be used to further reduce the bandwidth by up to 40% without further degradation of quality, in particular for applications where low latency is not essential (e.g. music streaming).

References

Pavel Andreev, Aibek Alanov, Oleg Ivanov, and Dmitry Vetrov. Hifi++: a unified framework for neural vocoding, bandwidth extension and speech enhancement. *arXiv preprint arXiv:2203.13086*, 2022.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

Bishnu S Atal and Suzanne L Hanauer. Speech analysis and synthesis by linear prediction of the speech wave. *The journal of the acoustical society of America*, 50(2B):637–655, 1971.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017.

Johannes Ballé, Nick Johnston, and David Minnen. Integer networks for data compression with latent-variable models. In *International Conference on Learning Representations*, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Dipesh Bhagat, Ninad Bhatt, and Yogeshwar Kosta. Adaptive multi-rate wideband speech codec based on celp algorithm: architectural study, implementation & performance analysis. In *2012 International Conference on Communication Systems and Network Technologies*, pp. 547–551. IEEE, 2012.

Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The mtg-jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, United States, 2019. URL `http://hdl.handle.net/10230/42015`.

Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis. *arXiv preprint arXiv:2111.05011*, 2021.

Shlomo E Chazan, Lior Wolf, Eliya Nachmani, and Yossi Adi. Single channel voice separation for unknown number of speakers under reverberant and noisy settings. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3730–3734. IEEE, 2021.

Michael Chinen, Felicia SC Lim, Jan Skoglund, Nikita Gureev, Feargus O'Gorman, and Andrew Hines. Visqol v3: An open source production ready objective speech and audio metric. In *2020 twelfth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6. IEEE, 2020.

Cisco. Global - 2021 forecast highlights - cisco. `https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf`, 2021.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

Alexandre Defossez, Gabriel Synnaeve, and Yossi Adi. Real time speech enhancement in the waveform domain. *arXiv preprint arXiv:2006.12847*, 2020.

Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable model compression via pseudo quantization noise. *arXiv preprint arXiv:2104.09987*, 2021.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *Advances in Neural Information Processing Systems*, 31, 2018.

Martin Dietz, Markus Multrus, Vaclav Eksler, Vladimir Malenovsky, Erik Norvell, Harald Pobloth, Lei Miao, Zhe Wang, Lasse Laaksonen, Adriana Vasilache, et al. Overview of the evs codec architecture. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5698–5702. IEEE, 2015.

Harishchandra Dubey, Vishak Gopal, Ross Cutler, Sergiy Matusevych, Sebastian Braun, Emre Sefik Eskimez, Manthan Thakker, Takuya Yoshioka, Hannes Gamper, and Robert Aichner. Icassp 2022 deep noise suppression challenge. In *ICASSP*, 2022.

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30: 829–852, 2021.

Cristina Gârbacea, Aäron van den Oord, Yazhe Li, Felicia SC Lim, Alejandro Luebs, Oriol Vinyals, and Thomas C Walters. Low bit-rate speech coding with vq-vae and a wavenet decoder. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 735–739. IEEE, 2019.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 776–780. IEEE, 2017.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. *arXiv preprint arXiv:2202.09729*, 2022.

Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

D Griffin and Jae Lim. A new model-based speech analysis/synthesis system. In *ICASSP*, 1985.

Alexey Gritsenko, Tim Salimans, Rianne van den Berg, Jasper Snoek, and Nal Kalchbrenner. A spectral energy distance for parallel speech synthesis. *Advances in Neural Information Processing Systems*, 33: 13062–13072, 2020.

Andrew Hines, Jan Skoglund, Anil Kokaram, and Naomi Harte. Visqol: The virtual speech quality objective listener. In *IWAENC 2012; International Workshop on Acoustic Signal Enhancement*, pp. 1–4. VDE, 2012.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Technical Report 1502.03167, arXiv, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Tejas Jayashankar, Thilo Koehler, Kaustubh Kalgaonkar, Zhiping Xiu, Jilong Wu, Ju Lin, Prabhav Agrawal, and Qing He. Architecture for variable bitrate neural speech codec with configurable computation complexity. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 861–865. IEEE, 2022.

Xue Jiang, Xiulian Peng, Chengyu Zheng, Huaying Xue, Yuan Zhang, and Yan Lu. End-to-end neural speech coding for real-time communications. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 866–870. IEEE, 2022.

Biing-Hwang Juang and A Gray. Multiple stage vector quantization for speech coding. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pp. 597–600. IEEE, 1982.

Nal Kalchbrenner et al. Efficient Neural Audio Synthesis. In *ICML*, 2018.

Eugene Kharitonov, Ann Lee, Adam Polyak, Yossi Adi, Jade Copet, Kushal Lakhotia, Tu-Anh Nguyen, Morgane Rivière, Abdelrahman Mohamed, Emmanuel Dupoux, et al. Text-free prosody-aware generative spoken language modeling. *arXiv preprint arXiv:2109.03264*, 2021.

W Bastiaan Kleijn, Felicia SC Lim, Alejandro Luebs, Jan Skoglund, Florian Stimberg, Quan Wang, and Thomas C Walters. Wavenet based low rate speech coding. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 676–680. IEEE, 2018.

W Bastiaan Kleijn, Andrew Storus, Michael Chinen, Tom Denton, Felicia SC Lim, Alejandro Luebs, Jan Skoglund, and Hengchin Yeh. Generative speech coding with predictive variance regularization. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6478–6482. IEEE, 2021.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

Felix Kreuk, Adam Polyak, Jade Copet, Eugene Kharitonov, Tu-Anh Nguyen, Morgane Rivière, Wei-Ning Hsu, Abdelrahman Mohamed, Emmanuel Dupoux, and Yossi Adi. Textless speech emotion conversion using decomposed and discrete representations. *arXiv preprint arXiv:2111.07402*, 2021.

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

Kushal Lakhotia, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, et al. On generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*, 9:1336–1354, 2021.

Ann Lee, Peng-Jen Chen, Changhan Wang, Jiatao Gu, Xutai Ma, Adam Polyak, Yossi Adi, Qing He, Yun Tang, Juan Pino, et al. Direct speech-to-speech translation with discrete units. *arXiv preprint arXiv:2107.05604*, 2021a.

Ann Lee, Hongyu Gong, Paul-Ambroise Duquenne, Holger Schwenk, Peng-Jen Chen, Changhan Wang, Sravya Popuri, Juan Pino, Jiatao Gu, and Wei-Ning Hsu. Textless speech-to-speech translation on real data. *arXiv preprint arXiv:2112.08352*, 2021b.

Shuyang Li, Huanru Henry Mao, and Julian McAuley. Variable bitrate discrete neural representations via causal self-attention. In *2nd Pre-registration workshop (NeurIPS 2021), Remote.*

Yunpeng Li, Marco Tagliasacchi, Oleg Rybakov, Victor Ungureanu, and Dominik Roblek. Real-time speech frequency bandwidth extension. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 691–695. IEEE, 2021.

Felicia SC Lim, W Bastiaan Kleijn, Michael Chinen, and Jan Skoglund. Robust low rate speech coding based on cloned networks and wavenet. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6769–6773. IEEE, 2020.

Ju Lin, Kaustubh Kalgaonkar, Qing He, and Xin Lei. Speech enhancement for low bit rate speech codec. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7777–7781. IEEE, 2022.

Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8):1256–1266, 2019.

Alan McCree, Kwan Truong, E Bryan George, Thomas P Barnwell, and Vishu Viswanathan. A 2.4 kbit/s melp coder candidate for the new us federal standard. In *ICASSP*, 1996.

Shigeo Morishima, H Harashima, and Y Katayama. Speech coding based on a multi-layer neural network. In *IEEE International Conference on Communications, Including Supercomm Technical Sessions*, pp. 429–433. IEEE, 1990.

Eliya Nachmani, Yossi Adi, and Lior Wolf. Voice separation with an unknown number of multiple speakers. In *International Conference on Machine Learning*, pp. 7164–7175. PMLR, 2020.

Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoit Sagot, Abdelrahman Mohamed, et al. Generative spoken dialogue language modeling. *arXiv preprint arXiv:2203.16502*, 2022.

Ahmed Omran, Neil Zeghidour, Zalán Borsos, Félix de Chaumont Quitry, Malcolm Slaney, and Marco Tagliasacchi. Disentangling speech from surroundings in a neural audio codec. *arXiv preprint arXiv:2203.15578*, 2022.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Richard Clark Pasco. *Source coding algorithms for fast data compression.* PhD thesis, Stanford University CA, 1976.

Adam Polyak, Yossi Adi, Jade Copet, Eugene Kharitonov, Kushal Lakhotia, Wei-Ning Hsu, Abdelrahman Mohamed, and Emmanuel Dupoux. Speech resynthesis from discrete disentangled self-supervised representations. *arXiv preprint arXiv:2104.00355*, 2021.

Sravya Popuri, Peng-Jen Chen, Changhan Wang, Juan Pino, Yossi Adi, Jiatao Gu, Wei-Ning Hsu, and Ann Lee. Enhanced direct speech-to-speech translation using self-supervised pre-training and data augmentation. *arXiv preprint arXiv:2204.02967*, 2022.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3454–3463, 2019.

Jorma Rissanen and Glen Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, 1981.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

B Series. Method for the subjective assessment of intermediate quality level of audio systems. *International Telecommunication Union Radiocommunication Assembly*, 2014.

Jan Skoglund and Jean-Marc Valin. Improving opus low bit rate quality with neural speech synthesis. *arXiv preprint arXiv:1905.04628*, 2019.

Marco Tagliasacchi, Yunpeng Li, Karolis Misiunas, and Dominik Roblek. Seanet: A multi-modal speech enhancement network. *arXiv preprint arXiv:2009.02095*, 2020.

Jean-Marc Valin and Jan Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5891–5895. IEEE, 2019a.

Jean-Marc Valin and Jan Skoglund. A real-time wideband neural vocoder at 1.6 kb/s using lpcnet. *arXiv preprint arXiv:1903.12087*, 2019b.

Jean-Marc Valin, Koen Vos, and Timothy Terriberry. Definition of the opus audio codec. *IETF, September*, 2, 2012.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

A Vasuki and PT Vanathi. A review of vector quantization techniques. *IEEE Potentials*, 25(4):39–47, 2006.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of Neural Information Processing Systems*, 2017.

Bernard Widrow, Istvan Kollar, and Ming-Chang Liu. Statistical theory of quantization. *IEEE Transactions on instrumentation and measurement*, 45(2):353–361, 1996.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020a.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020b.

Jaeseong You, Dalhyun Kim, Gyuhyeon Nam, Geumbyeol Hwang, and Gyeongsu Chae. Gan vocoder: Multi-resolution discriminator is all you need. *arXiv preprint arXiv:2103.05236*, 2021.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.